LED Blink

LEDs are small, powerful lights that are used in many different applications. To start, we will work on **blinking an LED, the Hello World of microcontrollers**.

It is as simple as turning a light on and off. Establishing this important baseline will give you a solid foundation as we work toward more complex experiments.

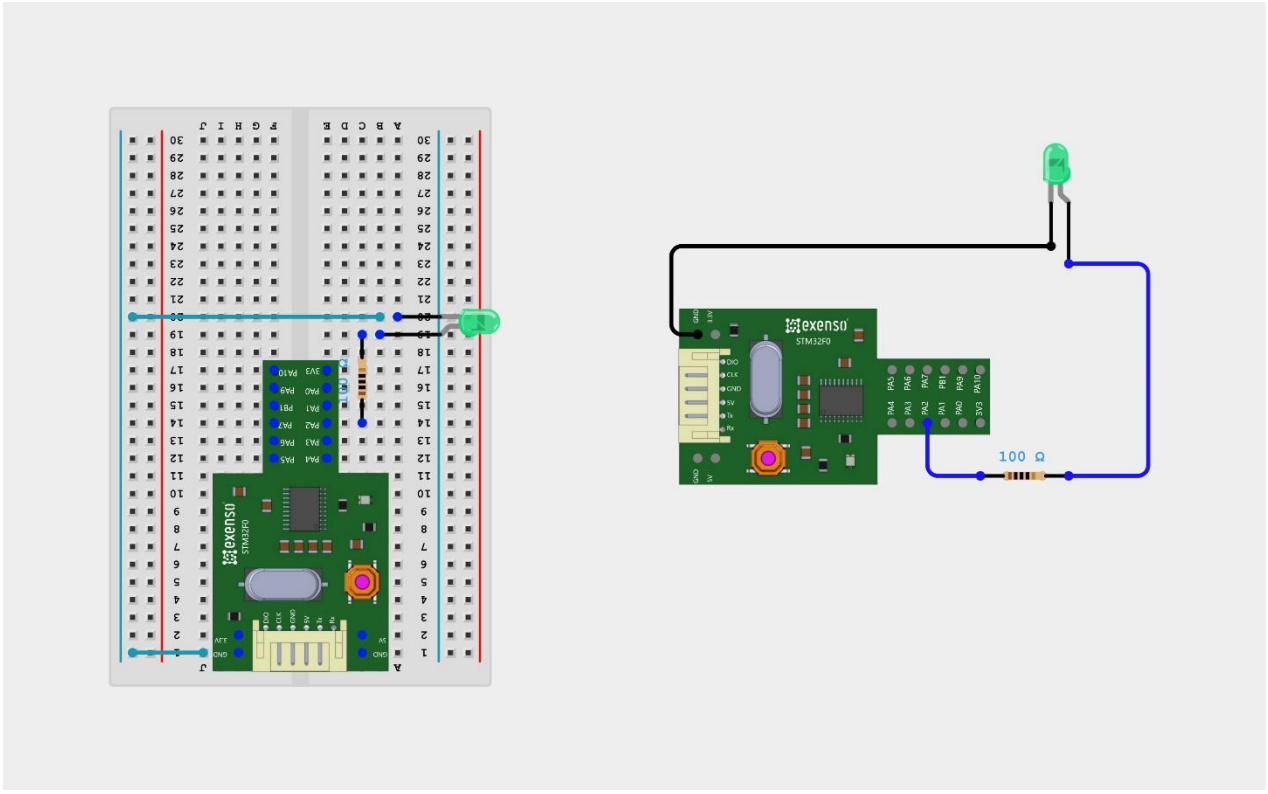
Components Required

You will need the following components −

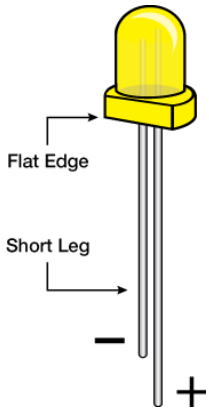
- 1 × Breadboard
- 1 × STM32F030F4P6
- 1 × LED
- 1 × 100Ω Resistor
- 2 × Jumper

Procedure

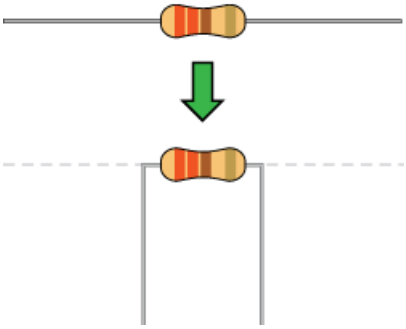
Follow the circuit diagram and hook up the components on the breadboard as shown in the image given below.



Note – To find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb indicates the negative terminal.



Components like resistors need to have their terminals bent into 90° angles to fit the breadboard sockets properly. You can also cut the terminals shorter.



Create STM32 Cube Ide new project

Open Cube Ide the following Figure 1

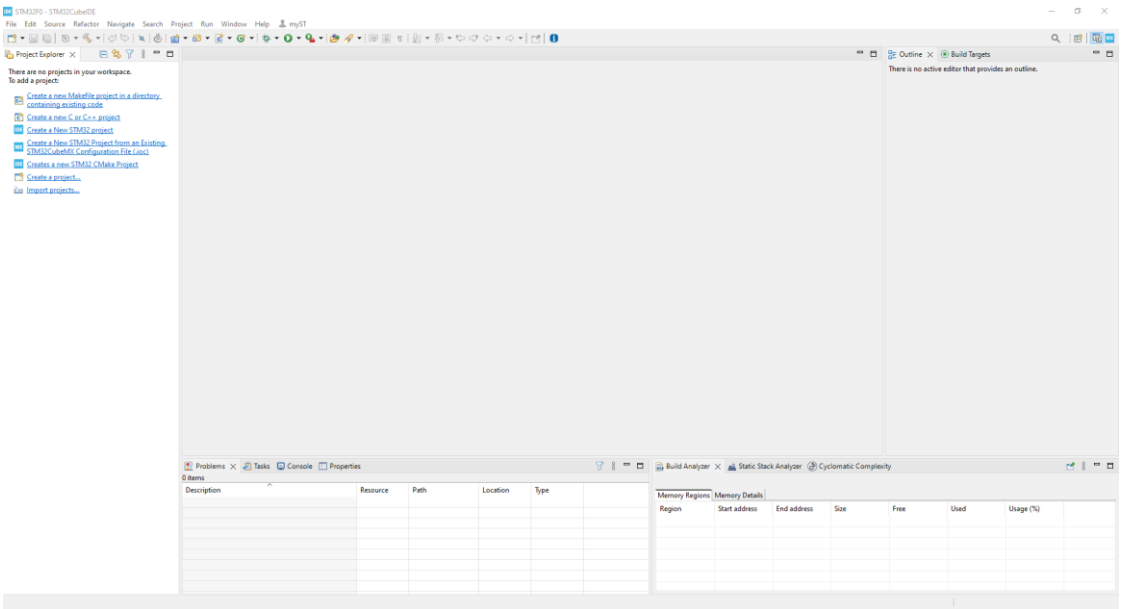


Figure: 1a

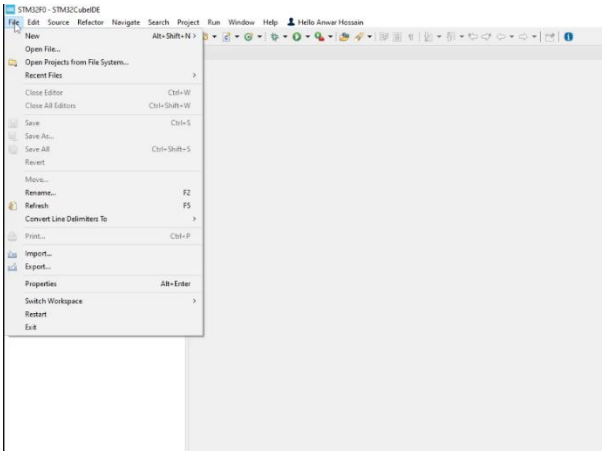


Figure: 1b

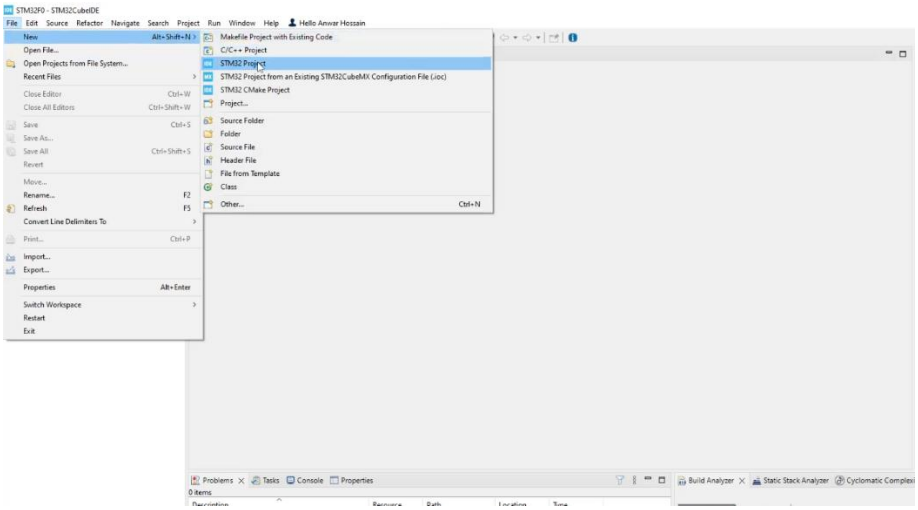


Figure: 1c

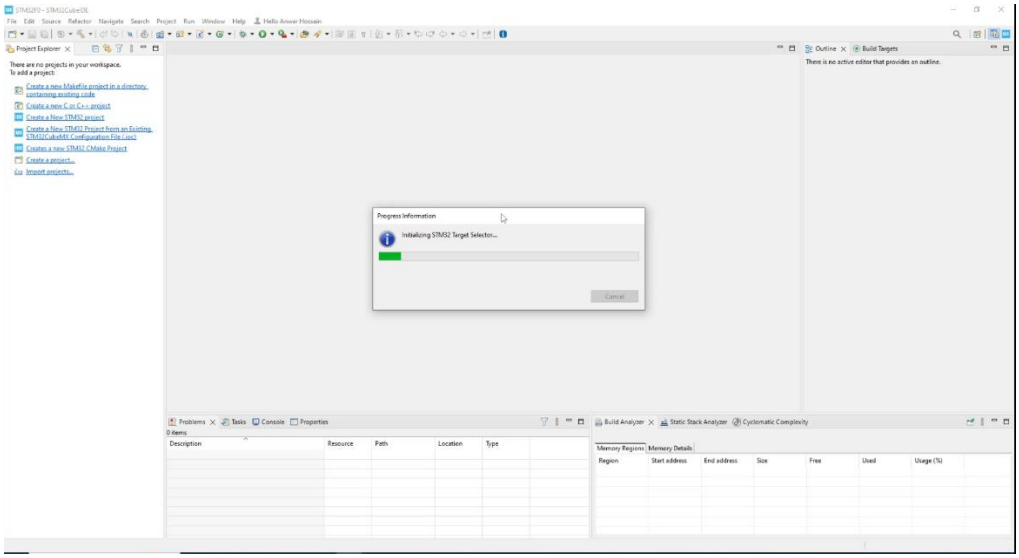


Figure: 1d

Select Board and Save the project following Figure 2

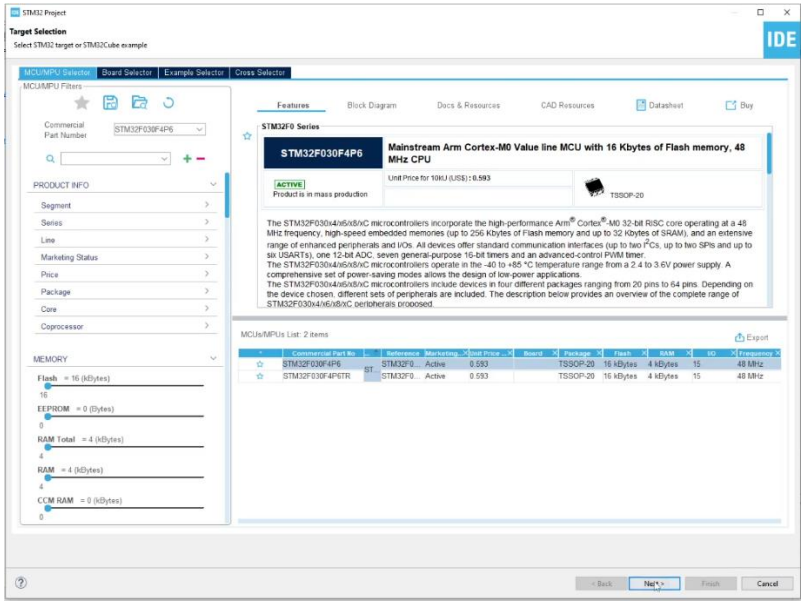


Figure: 2a

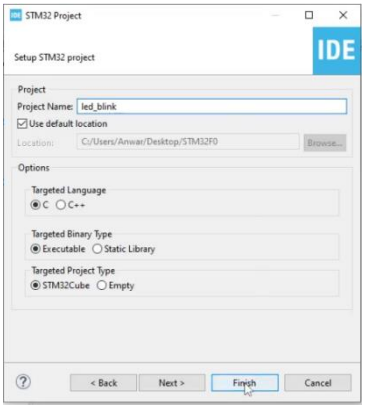


Figure: 2b

Pinout & Configuration window shows **PA2 (PortA Pin 2) Pin set as OUTPUT** and following step by step Figure 3

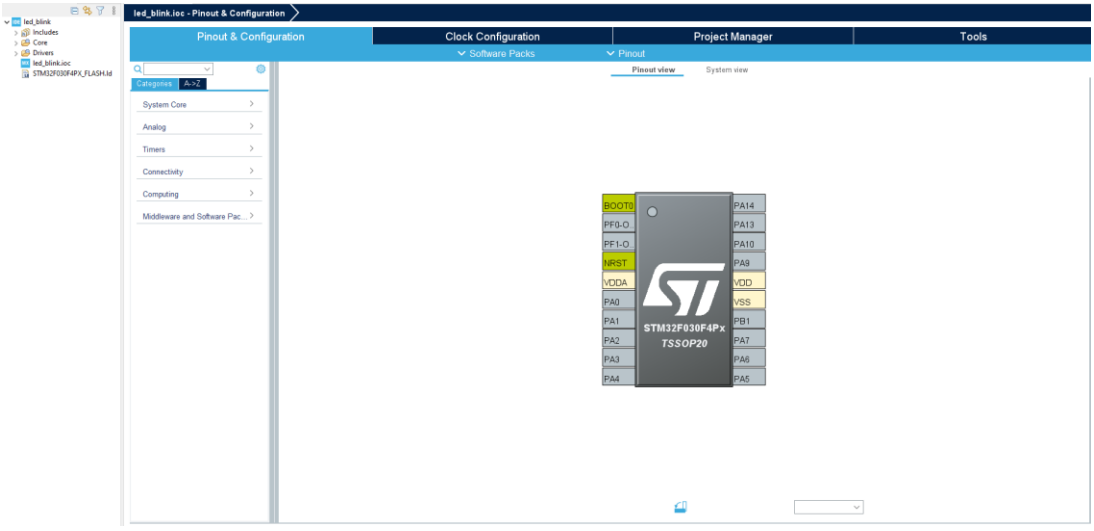


Figure: 3a

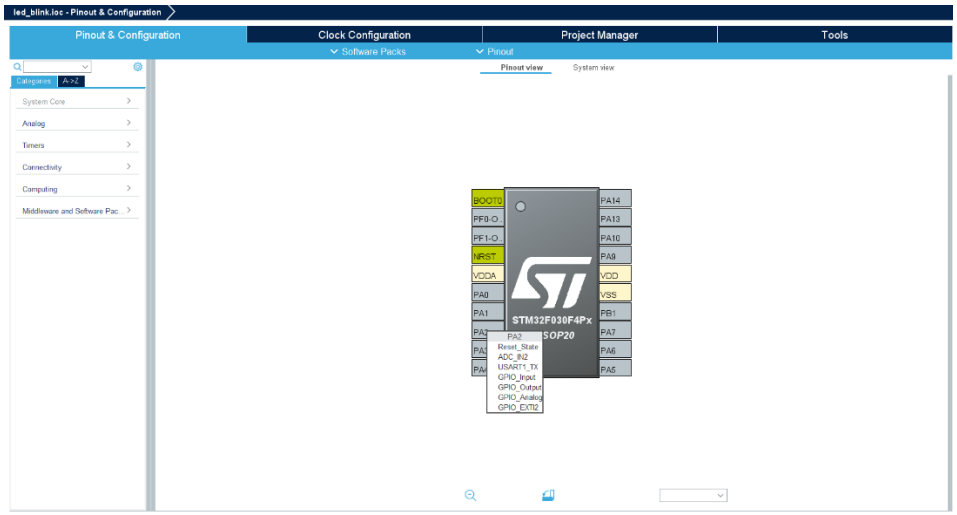


Figure: 3b

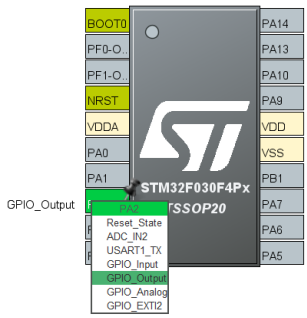


Figure: 3c

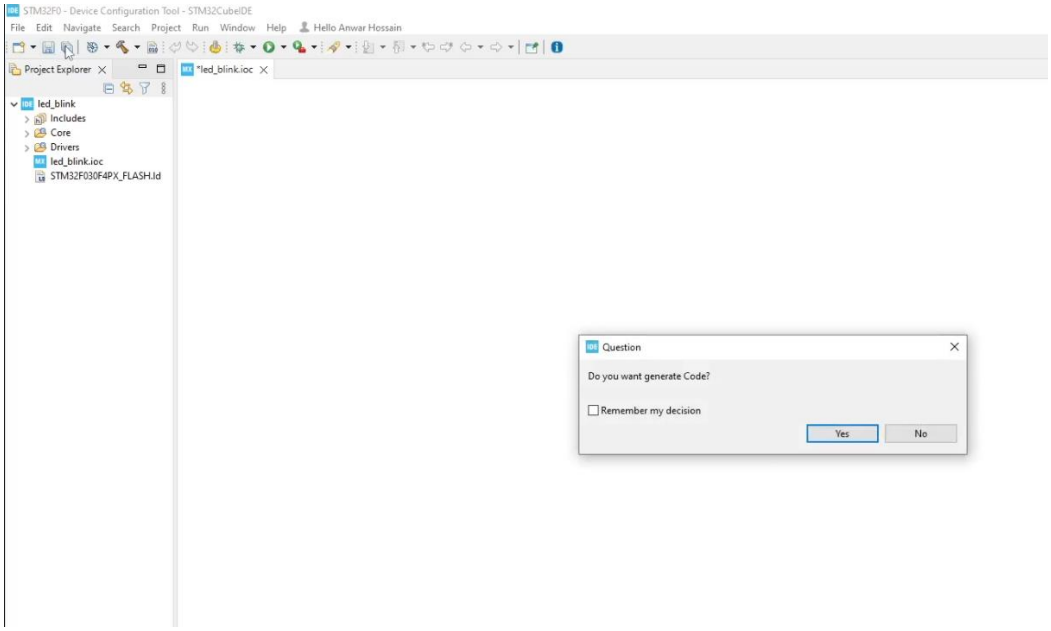


Figure: 3d

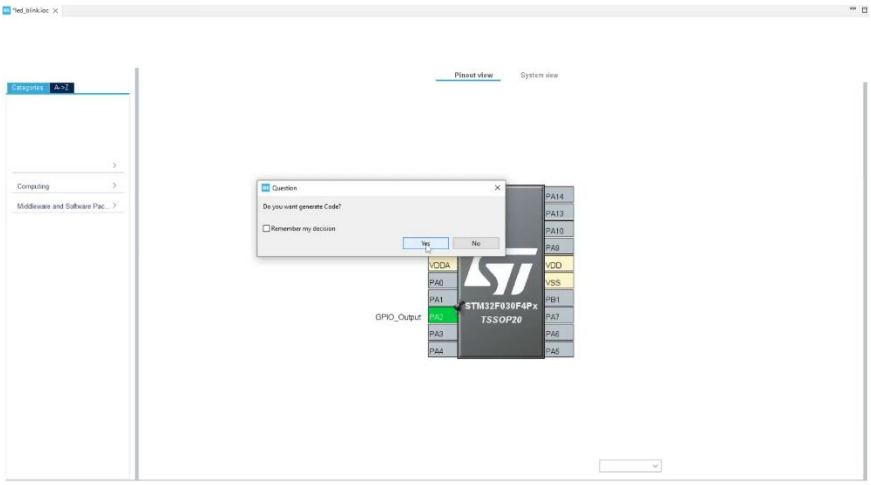


Figure: 3e

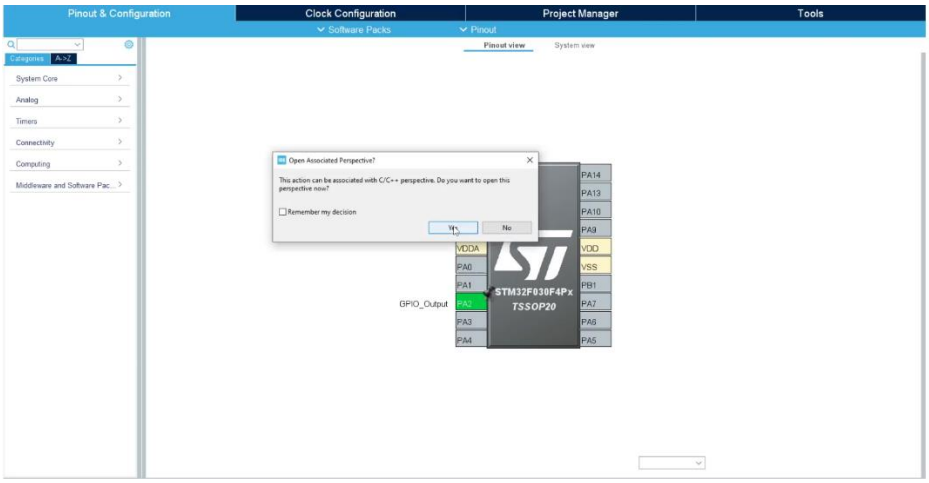


Figure: 3f

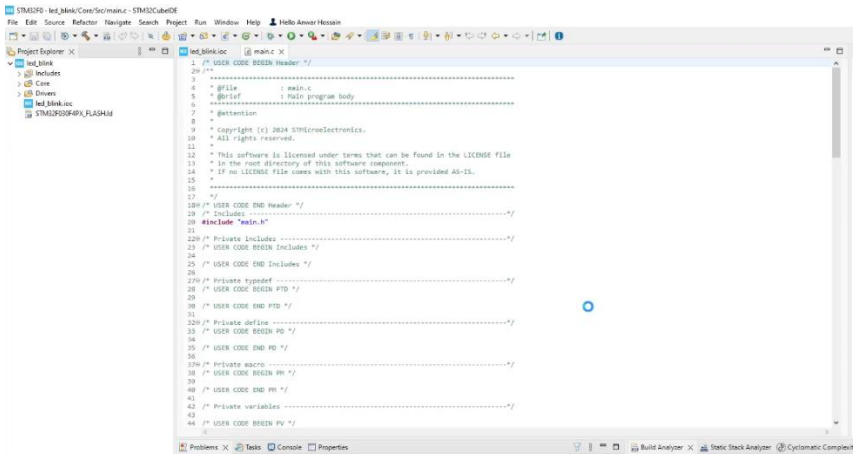


Figure: 3g

Code:

```
#include "main.h"

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  while (1)
  {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
        HAL_Delay(1000);   /* Insert delay 1 s */



  }
}
```

```
HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
HAL_Delay(1000);   /* Insert delay 1 s */
```

‘ HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2); ‘

- This line toggles the state of pin 2 of GPIO port A. If the pin is currently high (1), it will be set low (0), and vice versa. This function is part of the HAL library and provides a convenient way to toggle GPIO pins without explicitly checking their current state.

‘ HAL_Delay(1000); ‘

- Similar to the previous explanation, this line introduces a delay of 1000 milliseconds (1 second) using the HAL_Delay function. It's a simple delay function provided by the HAL library, causing the program to pause execution for the specified duration.

Another way blink led

```c
#include "main.h"


void SystemClock_Config(void);
static void MX_GPIO_Init(void);
int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  while (1)
  {
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_2,GPIO_PIN_SET);   // PA2 PinState is HIGH(1)
        HAL_Delay (1000);                                      /* Insert delay 1 s */
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_2,GPIO_PIN_RESET); // PA2 PinState is HIGH(1)
        HAL_Delay (1000);                                      /* Insert delay 1 s */
  }

}
```

```c
       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2,GPIO_PIN_SET);   // PA2 PinState is HIGH (1)
        HAL_Delay(1000);                                     /* Insert delay 1 s */
        HAL_GPIO_WritePin (GPIOA, GPIO_PIN_2,GPIO_PIN_RESET); // PA2 PinState is HIGH (1)
        HAL_Delay (1000);                                     /* Insert delay 1 s */
```

‘ HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2,GPIO_PIN_SET); ‘

- This line is setting pin 2 of GPIO port A to a high state (logic 1). GPIOA likely refers to a GPIO port, and GPIO_PIN_2 refers to pin 2 within that port. GPIO_PIN_SET is a macro or enum constant indicating to set the pin **high**.

‘ HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2,GPIO_PIN_RESET); ‘

- This line is resetting pin 2 of GPIO port A to a low state (logic 0). GPIO_PIN_RESET is a macro or enum constant indicating to reset the pin low.

‘ HAL_Delay(1000); ‘

- This line introduces a delay of 1000 milliseconds (1 second) using the HAL_Delay function. It's a simple delay function provided by the HAL library.

When Code writing is done then compile the code and upload the program on the STM32F0 dev board following Figure 4

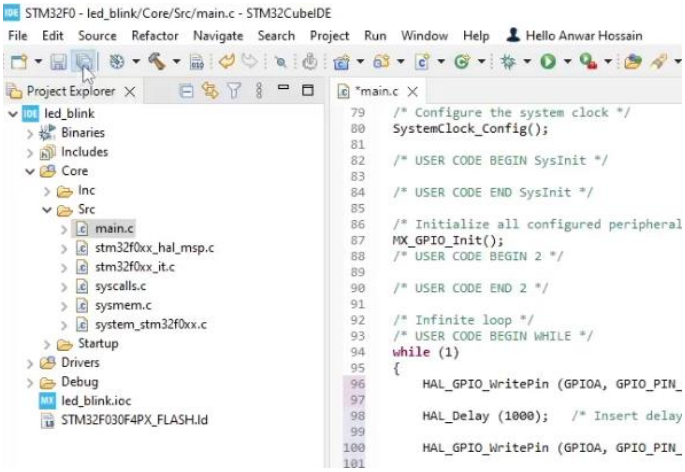


Figure: 4a

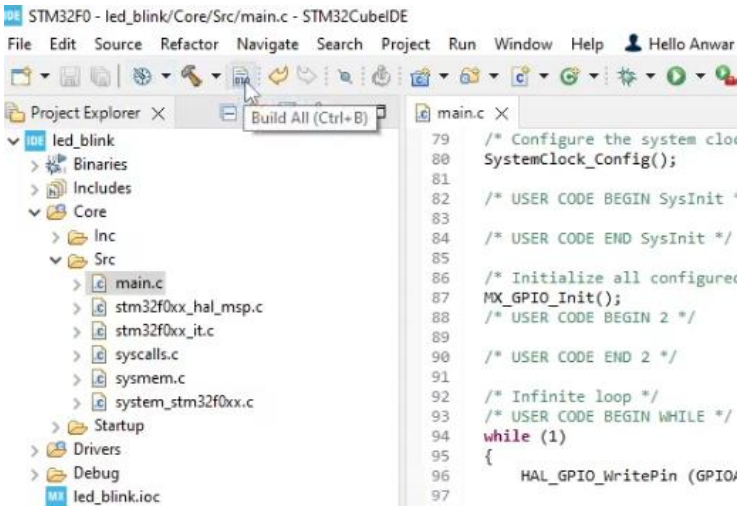


Figure: 4b

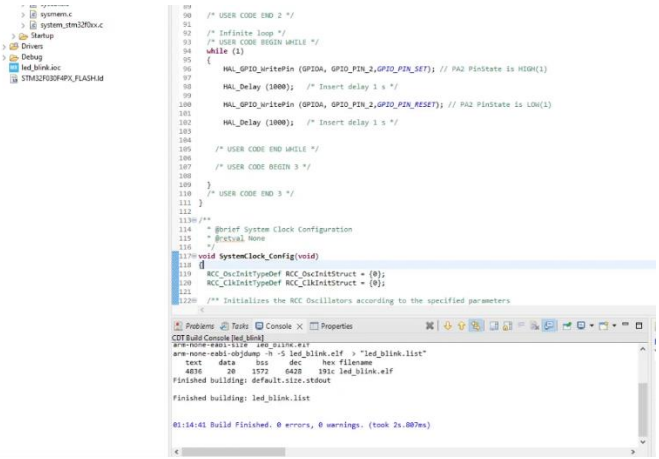


Figure: 4c

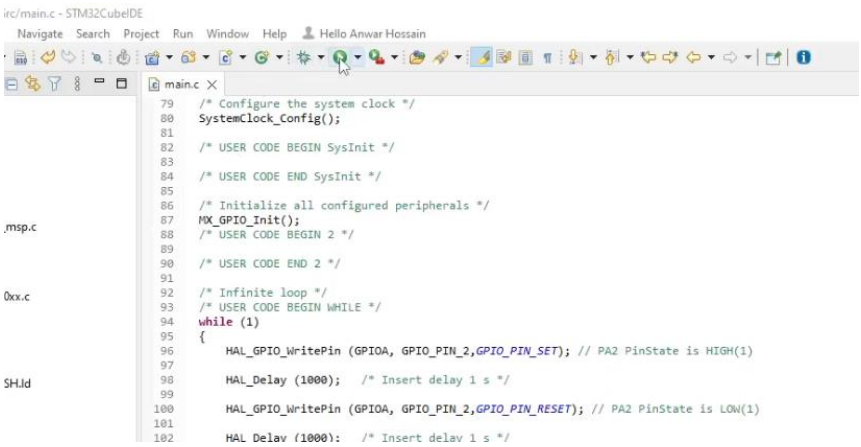


Figure: 4d

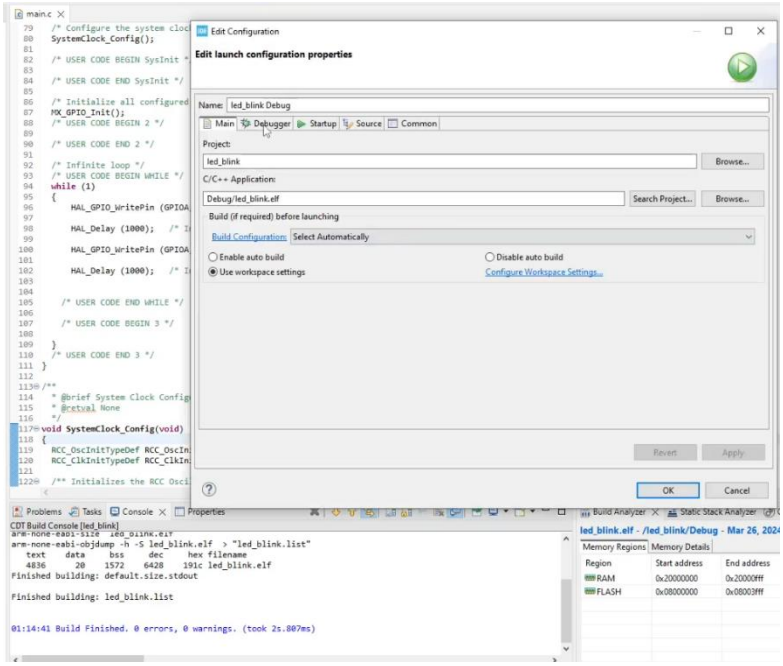


Figure: 4e

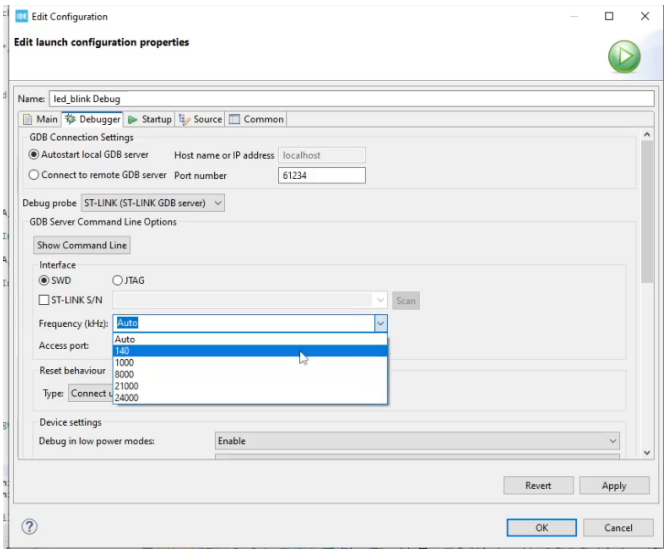


Figure: 4f

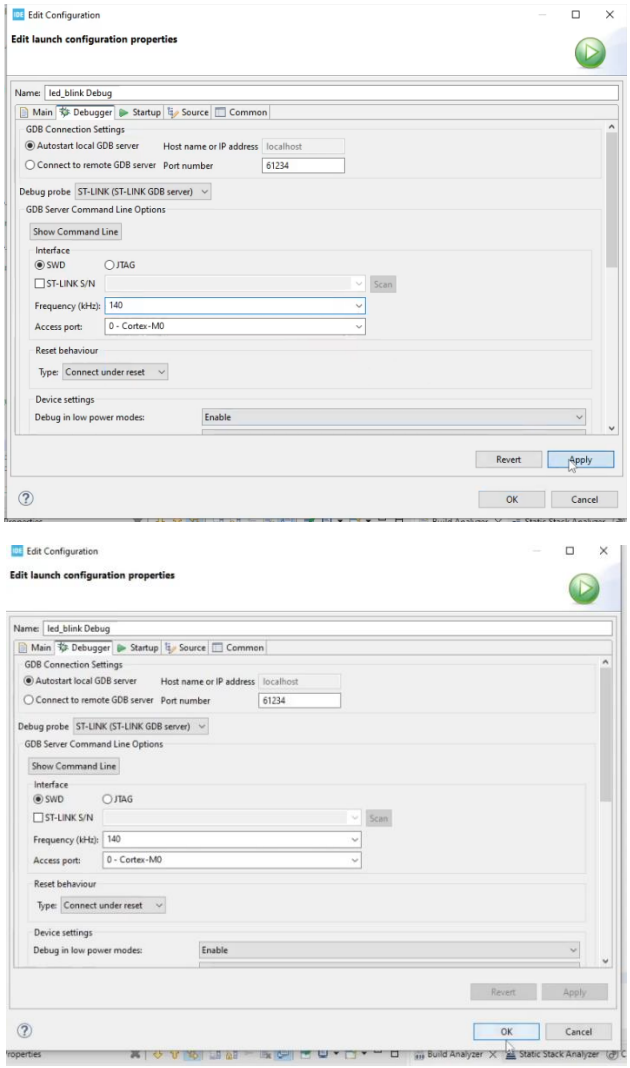


Figure: 4g

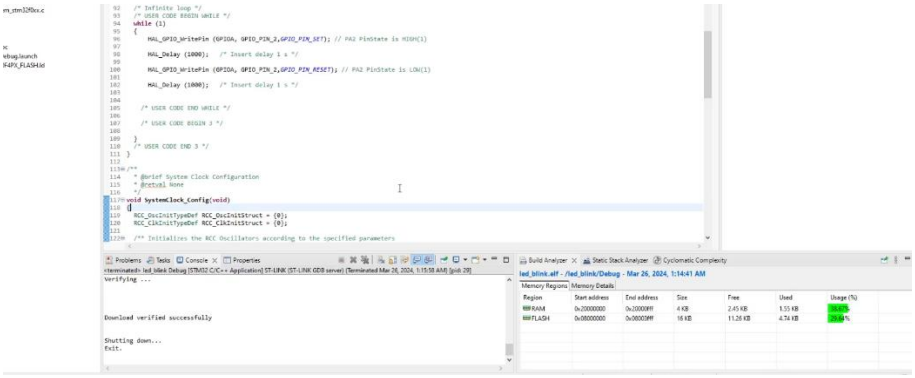


Figure: 4h